DATA RETRIEVAL METHOD AND APPARATUS WITH MULTIPLE SOURCE CAPABILITY

The present application is a continuation-in-part of application serial number 08/593,118, filed February 1, 1996, which is a continuation-in-part of application serial number 08/582,062, (attorney file number 23134/90100) filed December 30, 1995 for DATA RETRIEVAL METHOD AND APPARATUS WITH MULTIPLE SOURCE CAPABILITY, both of which are incorporated herein by reference.

The present invention relates to a computer-implemented system which is able to retrieve information stored in one or more of a number of different sources and which may be in any of a number of different formats and/or provide reports and analysis based on the information, and in particular to a computer method and apparatus which can automatically retrieve database information stored in any of a plurality of formats, including structural and/or relational information, without the need for relying on human analysis of the source data.

BACKGROUND INFORMATION

A number of ways of organizing computer-accessible information have developed, such as relational or hierarchical database management systems, flat file data systems, spreadsheet systems, and the like. These systems are used for storing, manipulating and displaying a myriad of types of information, including accounting or other financial information, scientific or technical data, corporate or business data, name, address and telephone data and statistical data. Many formats and data structures have been developed, and this situation has both desirable and undesirable ramifications. On the positive side, by having a multiplicity of different types of systems, it is possible to provide different systems which are optimized for different purposes (e.g., optimized for data entry or storage vs. speed or flexibility of data analysis and reporting, optimized for accounting data vs. company data, and the like), or which provide user interfaces or other characteristics which may appeal to personal or company preferences. This multiplication of information systems, however, provides a substantial barrier in situations in which it would be useful to have access to information in two or more such systems, e.g. to coordinate or combine such information. Examples of such situations include: (1) an accountant who wishes to produce standardized reports but who has multiple clients, each of whom keeps its accounting data in a different type of data source; (2) a corporation with several divisions which wishes to produce uniform reports, but in which different divisions use different corporate or financial software; (3) a corporation which wishes to produce uniform reports, but which keeps its accounting information on a first type or brand of database (or other data source), and its corporation information on a second and different type of database; (4) a group of scientists investigating a common problem, each of whom stores or has access to data kept in a different type or brand of database or other data source. Other examples will occur to the reader after understanding the present disclosure. Additionally in some situations, even when all the desired information is stored in a single type of data source or in a single data file, it may be desirable to provide a manner of accessing the data, e.g., to provide for uniform and/or enhanced reporting and analysis of the data.

Such situations present difficulties for a number of reasons, including the difference in manners of organizing information and differences between types of data sources. In some situations, similar categories of information may be organized in different ways, even if the same database software is being used. For example, in

a first instance, using a first database software package, a user might organize a company's personnel records such that all of the company's personnel names are stored in a first table or list, all of the addresses are stored in a second table or list, and all of the telephone numbers are stored in a third table or list, and pointers or links are stored to indicate which names are associated with which addresses and which phone numbers. However, another instance using the same software might occur in which a different person organizing personnel information might provide a single table in which each line or "record" of information includes a name, an address and a telephone number, thus without any links or pointers from a record in one table to a record to another table.

Additionally, different types of data sources may have different structures and/or different data storage formats or schemes. For example, some database packages are organized in a hierarchical manner (e.g., in a tree-fashion), while others may be organized as relational databases (modeled on two-dimensional tables of rows and columns). Furthermore, information may be stored in forms that are not, strictly speaking, database forms such as storing data in a "flat file" form, as a spreadsheet, and the like. Additionally, different types of data sources may store the data in various formats. For example, some database products store each table, each reporting format and each query as a separate file on a storage device such as a hard disk, while other software may store all tables, relationships, queries, report formats, etc., in a single file. Some products may store each record and/or field as fixed length data and/or at a fixed position in a file, while others may use delimiters to distinguish between one record and the next or between one field and the next within a record. Even if two different software products store a particular type of information at a predetermined location, such location may be different for the different software products. Furthermore, data may be encoded differently in different software products, such as using ASCII encoding in one product and multi-lingual (multi-byte) characters in another product. In some cases, data may be compressed and/or encrypted.

In view of the wide variation among types of data, in the past, when it was desired to access stored information (e.g. to standardize reports and analysis and/or to combine or coordinate information from two or more databases), a consultant or other expert individually or "manually" analyzed each "source" data file or database to understand its structure, relationship data storage format, the organization of the data within the database, and the like. The expert would then construct some manner of import or querying of the data in the source data file or database in order to achieve the desired access, coordination or combination. Although this approach is operable, it is labor-intensive, since it requires human analysis, and is also time-consumptive since a relatively long period of time is typically required for the expert or consultant to complete the task of analyzing, often requiring days or weeks for the access, coordination or combination to be achieved.

Accordingly, it would be useful to provide a system in which information in various formats or forms or organized in various ways can be accessed combined and/or coordinated, while reducing or eliminating the need for human analysis, thus providing a system which is at least partially automated and preferably less labor-intensive and less time-consumptive than certain previous methods.

SUMMARY OF THE INVENTION

The present invention relates to a system which achieves access to stored information, e.g., for accessing information or for achieving coordination and/or combination of information in two different information storage

systems. Preferably, some or all the analysis involved is performed automatically (i.e., without the need for human analysis), in one embodiment, using a properly programmed computer. Preferably, the system is flexible in that it is not inherently limited in the data formats it can access but can be configured to obtain data from virtually any computer-readable information source. Preferably the system is extensible, (more preferably, modularly extensible) in that components can be added to permit it to access additional types, formats or organizations of data. In one embodiment, the access, coordination or combination of data is accompanied by an enhancement of data analysis, i.e., providing types of data analyses and/or reporting not found or used in the original data source. Preferably, the system can be used to provide for standardization of data analysis or reporting across several types of data sources. In one embodiment, the system uses the contents of the source data files or databases, as well as information about the structure, in order to achieve the desired results (such as by using text recognition, artificial intelligence, and/or expert systems). In one embodiment, the system uses such information to at least partially control the manner in which data is made available for analysis or reporting. In one embodiment, the system uses such information in providing such analysis or reports.

Generation of output or reports on information contained in a data source which may be any of two or more types of source data, in a standardized or uniform manner is provided. A plurality of drivers are provided specific to different types of source data which include programming for identifying structural or other characteristics of the various data sources, e.g. for use in defining a new database. Preferably the new database is configured to permit highly flexible and/or rapid output or reporting or is otherwise optimized for reporting purposes. In one embodiment, the present invention includes conversion of one or more data sources into one or more uniform databases, preferably generating one or more key categories for organizing the data, optionally generating category groupings or rollups and additional data or optional references.

In one embodiment, the present invention creates or populates a database, based on accounting or other data converted from existing data files, such as data files created by previous accounting or other software.

In one embodiment, the system is configured to facilitate updating some or all portions of the new database such as by storing one or more profiles of data which should be updated and/or creating or defining a schedule to automatically execute update procedures at predefined times or intervals.

In one embodiment, enhancements are provided which automatically identify a desirable view of the data such as by generating a view when more than one value is found for a given field or parameter of the data.

Preferably, one or more verification or audit tools are provided to detect potential errors or problems.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic depiction of an example of flat file data storage;

Fig. 2 depicts a directory structure of a type usable in connection with the data storage depicted in Fig. 1;

Figs 3A - 3C depict examples of data storage formats usable in connection with the data storage depicted in Fig. 1;

Figs. 4A - 4F are schematic depictions of an example of data stored in tables of a relational database ;

Fig. 5 depicts a directory structure of a type usable in connection with the data storage depicted in Figs 4A - 4F;

Fig. 6 is a schematic depiction of an example of flat file data storage;

Figs. 7A - 7D are schematic depictions of data stored in tables of a relational database;

Fig. 8 is a block diagram of a system for information retrieval according to an embodiment of the present invention;

Fig. 9 is a schematic depiction of the contents of a function module according to an embodiment of the present invention;

Fig. 10 is a flowchart of a process for information retrieval according to an embodiment of the present invention;

Figs. 11A and 11B depict pseudocode procedures for selecting or searching directories, according to an embodiment of the present invention; and

Fig 12 is a schematic depiction of data stored in tables of a database 808 provided according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Before describing certain aspects of the present invention, it will be useful to promote an understanding of the present invention, to provide examples of various fashions of storing information. This will be done by providing several examples, including examples of accounting information and examples of scientific or technical information. Table I provides a comparison of types of data that might be stored by two different corporations. Table I is intended to indicate the conceptual organization of accounting and other information for two corporations, and is not necessarily information that would be stored in a database (although it could be, if desired).

Table I. Examples of Accounting Organization

Corporation # 1	Corporation # 2
Account	Account
Cash	Cash
Accounts Payable	Bank #1
Accounts Receivable	Bank #2
	Bank #3
	Savings
	Checking
	Accounts Payable
	Parts
	Consultants
	Accounts Receivable
	Sales
	Interest

Table I. Examples of Accounting Organization

Corporation # 1	Corporation # 2
Employee	Employee
Salesman #1	Sales
Salesman #2	Salesman #1
	Salesman #2
	Research
	Researcher #1
	Researcher #2
Project	Project
Research	Research
Sales	Chemical
	Biomedical
	Sales
	Old Products
	New Lines
Product	Product
Product #1	Old Product Lines
Product #2	Line #1
	Product 1
	Product 2
	Line #2
	Product 3
	Product 4
	New Product Lines
	Line #3
	Product 5
	Product 6
	Line #4
	Product 7
	Product 8
	Subsidiary
	Subsidiary #1
	Subsidiary #2

In the first example of Table I, corporation no. 1 retains Account information, employee information, project information and product information and thus has four account parts. The Account information of corporation no. 1 includes only three components: cash, accounts payable, and accounts receivable. The corporation maintains a list of its salesmen, tracks information by two types of projects (research and sales) and maintains a list of its products. The second example of Table I is a somewhat lengthier (but still simplified) example. In this example, the Account has multiple components in a hierarchy. Although corporation no. 2's account has categories of cash, accounts payable and accounts receivable, each of these categories has subcategories and some of the subcategories have even finer divisions. Similarly, the employee, project and product information is broken into several categories and subcategories, and corporation no. 2 further has additional items which are tracked, such as subsidiary companies.

Table I illustrates that, even before considering differences among data sources and/or data storage formats, a corporation's structure and/or the manner in which it chooses to organize its information will cause differences from one system to another. For example, if an accountant had corporation no. 1 and corporation no. 2 as clients and wished to use a uniform or standard system of reporting and analysis or for these two corporations, there would be significant difficulties in doing so, even if corporation no. 1 and corporation no. 2 used the same database software, and even organized accounting information within that software in a similar fashion. Thus, using previous methods, it typically would have required human analysis and understanding of the information in Table I in order to provide uniform or standardized reporting and analysis for the two corporations, based on those corporations' databases.

Still in the context of considering accounting information, several types of information storage may be used for storing the accounting information of corporation no. 1 and/or corporation no. 2. For example, the information may be stored as one or more flat files. It is noted that, at least according to some usage, "flat file" information storage is not a true database system. Nevertheless, the present invention, in at least some embodiments, is able to accommodate flat file data, as well as other database and non-database storage methods.

Fig. 1 is a schematic depiction of how a plurality of flat files might be used to store information for corporation no. 2 of Table I. Although Fig. 1 shows the information as it might appear in written form on a plurality of sheets of paper, the data will, in fact, be stored on a computer readable medium such as a hard disk, e.g., as described below. The format of Fig. 1 is intended to depict the logical structure of the data organized into a plurality of files 101a through 101f, each file having file identification information 104, depicted in Fig. 1 as title or header information 104a, 104b, and a plurality of records, depicted in Fig. 1 as lines of information 106a, 106b, 106c, each record having a plurality of fields (organized into columns 108a, 108b, 108c, 108d in the depiction of Fig. 1). Methods and apparatus for storing and accessing data to have or reflect the logical row and column structure depicted in Fig. 1 are well known to those of skill in the programming art. The present invention can be used in connection with a number of flat file information storage programs. Examples of such information storage programs include those sold under the tradenames, Simply Accounting™ and MAS-90™.

The data organized in the logical structure depicted in Fig. 1 may be stored in a number of different formats. For example, in one embodiment the data in each flat file 101a through 101f, is stored in a separate file on

a hard disk of a personal computer. Fig. 2 depicts a directory/file structure which might be used for storing such files, in which all files depicted in Fig. 1 are stored in a single subdirectory. As is well known to those of skill in the art, the various files, even though organized in a directory hierarchy as depicted in Fig. 2, may be stored, physically, on a hard disk in a plurality of separated locations. A number of formats may be used for storing the data in a file. Examples are depicted in Figs. 3A through 3C. In the example of Fig. 3A, the file includes header information, followed by storage of the first record 106a, the second record 106b, and so on. In the example of Fig. 3A, a fixed length data format is used in which each record 106a, 106b, has an identical length 304 (i.e., occupies a fixed number of bits). In the embodiment of Fig. 3A, each field within each record also has a fixed length 308a through 308d.

Fig. 3B depicts another fixed length data storage method in which data is stored in column order rather than row order (all dates sequentially, then all descriptions sequentially, etc.). In the embodiment of Fig. 3B, in order to assist in locating desired data, it may be useful to store an indication 322 of the number of records, e.g. as part of the header. In the fixed length system, e.g., as depicted in Fig. 3A or 3B, a particular piece of data will be located a given distance (i.e., a given number of bits) from the beginning of the data. For example, in Fig. 3A, if the header 302 is known to have a length of four bytes, and the record length 304 is known to be 8 bytes, the data information for the first record 106a will necessarily be found beginning at byte number 5, the data information with the second record 106b will necessarily be found beginning at byte number 13, and so forth.

Fig. 3C depicts storage of data in delimited fields rather than fixed-length fields. In delimited format, a special symbol, i.e. a pattern of bits which is different from any pattern used for storing data, is used to mark the end or beginning of a record and/or a field. In the embodiment depicted in Fig. 3C, two different special symbols are used, one to mark the beginning of a record and another to mark the beginning of a field. These symbols are indicated in Fig. 3C, respectively, as a colon and a semicolon, which may stand for any bit pattern or symbol. In the data format of Fig. 3C, it is possible to identify the information indicating the date information for the first record 106a as the information following the first new record symbol 324a, and prior to the first new field symbol 324b. The date information from the second record 106b would be that information following the second new record symbol 326a and preceding the next-following new field symbol 326b, etc. Many other formats for storing information are possible. It should be apparent from the illustrations of Figs. 3A through 3C that the multiplicity of data storage formats presents yet another problem to access, coordination and combination of data in different types of information storage systems. Previously, those wishing to access information directly (e.g., without using the database management system or other software intended to read the stored information) required knowledge of the data storage formats such as might, in some cases, be acquired by analyzing examples of the stored information. Thus, accommodating the needs of an accountant who wishes to access information stored for both corporation no. 1 and corporation no. 2 of Table I would require not only information about the logical organization of the data (Fig. 1), its logical directory structure (Fig. 2), but also information about data storage format (Figs. 3A through 3C).

In addition to storage of information as flat file data, a number of other possibilities exist. Figs. 4A through 4F depict one possible organization of information in the context of a relational database. In the example of Figs. 4A through 4F, a first transactions table 402 is stored, having a plurality of records 406a through 406d. It will

be noted that the records depicted in Fig. 4A are similar in some respects to those depicted in Fig. 1, i.e., include a data field, description field, quantity field, and amount field. In the example of Fig. 4A, an additional index field 408 is provided for each record. The transaction table in the example of Fig. 4A does not include the header information 104a of the type depicted in Fig. 1, and only a single transaction table Fig. 4A is provided (rather than a multiplicity of tables 101a through 101f in the embodiment of Fig. 1). In the relational database depicted in Figs. 4A through 4F, additional tables are provided which may reflect the organization depicted in Table I. For example, an Account table 412 includes a list of all the categories defined in Table I, with an index 414 being associated with each account. Similarly, a employee table 416 includes the names of the employee of Table I, each having an index 418 associated therewith. Also in Fig. 4C is an indication, for each name, of whether the person is associated with the sales force or the research force (reflecting the hierarchy shown in Table I). Furthermore, a field is included to indicate the location of the employee. Additional tables (not depicted) may be provided to list the various projects, products and subsidiaries of corporation no. 2, reflecting the organization of Table I.

Fig. 4D depicts a link table 422, which indicates, for each record of a transaction table 402, any desired links to other tables. For example, if the first transaction 406a is a transaction relating to the bank no. 1 component of the cash account, a record 428 will be provided, indicating that for the transaction record having index value no. 1, the appropriate account reference is that having the index no. 424. Similarly, links may be made to the employee table 416, or other tables (not shown). Thus, whereas in Fig. 1 it was necessary to provide a separate file for each possible combination of account, subsidiary, product, project, etc. (potentially leading to a very large number of files for a relatively complicated accounting structure), only a single transaction table 402 is required in the embodiment of Figs. 4A through 4F, with the link table Fig. 422 providing the information which, in the example of Fig. 1, is obtained by knowing in which flat file a transaction is stored.

In a typical relational database, it is possible to identify and display only information meeting certain criteria, e.g., only those transactions for a particular account and particular employee. In some database software it is possible to store such criteria or "queries" for use, e.g., when the selective information is commonly needed. Fig. 4E depicts a table which stores a plurality of such queries using, e.g., structured query language (SQL). The queries which are used in a particular database system may reflect the manner in which the data is analyzed or organized by a company. And thus, an accountant who may have an interest in standardizing reporting and analysis based on information in such a database may wish to know about and/or be able to reproduce the data analysis of the type represented by various stored queries (Fig. 4E).

Additionally, many types of database permit a user to design a report (either for display or printing) and, in some cases, to store information defining such a report, e.g. for repeated use. Thus, another table or set of tables (not shown) may be stored as part of or in connection with a relational database to preserve information regarding such reports.

The information and structure represented in Figs. 4A through 4F may be stored in a number of different fashions. Fig. 5 depicts a directory/file hierarchy which may be used for storing a plurality of data tables, link tables, query tables, and/or report formats. Such data may be stored in a number of different data formats, such as any of those depicted in Figs. 3A through 3C, or others, as will be known to those with skill in the art.

Another example of information which may be stored in various formats is scientific or technical information. In Fig. 6A, a flat file system is provided for storing surface temperature information for, e.g., meteorological research. In the example of Fig. 6, each file 602a, 602b, 602c stores information for a particular location and unit of measure (e.g., Fahrenheit or Celsius), indicated in the header thereof 604. For each record 606a, 606b, 606c the date and the reading for each hour of the day is stored in separate fields. A relational database system for storing this type of data is depicted in Figs. 7A through 7D. A data table 702 contains all of the observations and, for each observation, unit of measure 704a, along with an index 704a. Tables 706 (Fig. 7B) and 708 (Fig. 7C) store information relating to location and hours of the day. Each line 722, 724 of the link table (Fig. 7D) can be used to indicate, for any data point stored in Fig. 7A (as indicated by its index value 704b), the index value for the location associated with that data point and the time of day associated with that data point. Although the examples of Figs. 6 and 7A through 7D are simplified, it will be apparent to those of skill in the art how to construct information systems for storing more complicated sets of data such as meteorological data, including wind speed and wind direction, radio sound data, and the like. Thus, if a researcher wished to correlate information from two meteorological data sources, one of which stored information in the system depicted in Fig. 6 and another storing information in the system depicted in Fig. 7A through 7D, in previous methods it would typically be necessary to have a consultant or other expert analyze the information storage structure and organization depicted in Figs. 6 and 7A through 7D, and manually develop a system for accessing the information in such a fashion as to permit the combination or coordination of the data therein.

Fig. 8 is a block diagram of a system, according to one embodiment of the present invention, for use in overcoming the difficulties described above in relation to accessing, coordinating, or combining data on different information systems. In the embodiment of Fig. 8, a main process 802 selectively activates various drivers 804a, 804c, 804d, as needed in connection with analyzing and/or accessing information in a plurality of data sources 806a, 806b, 806c, 806d. For purposes of discussion, the source data 806a through 806d, depicted in Fig. 8, can be data which is stored or created by any of a number of programs or systems for organizing or storing data such as flat files systems, databases, spreadsheets, etc., as discussed above. The processes and data of Fig. 8 may reside on one or many computers. In one embodiment, the process is implemented in the context of a local-area network (LAN), having a network server computer, and associated disk or other storage devices, and one or more client computers. In one embodiment, the main process 802 is executed on a client computer while the information sources 806 and the data files for the new database 808 are stored on disk (or other data storage) associated with the network server. The invention can be implemented in many other architectures, such as on a single computer, on multiple non-networked computers, using computers which are linked by a communications link such as a wide area network, modem communications, via the Internet, etc. A number of types of computers can be used to implement the invention, including mainframe and personal computers such as those produced by International Business Machines (e.g. 386, 486 or Pentium or Pentium Plus based computers), Apple Corp. (such as MacIntosh computers) and clones of such computers. In one embodiment, the processes are implemented using a DOS operating system and/or a Microsoft® Windows or Windows 95 user interface.

The items referred to herein as drivers 804a through 804d, should not be confused with a data filter of a type provided in certain database programs. A data filter is, typically, a type of query or logical test for selecting

certain records and/or fields, according to user determinable criteria. The drivers 804a through 804d, on the other hand, as described more thoroughly below, are processes having multiple functions for analyzing and accessing different types of source data. In one embodiment, the function modules 804 are provided as dynamic link libraries (DLLs) in a manner that will be understood by those of skill in the art after understanding the present disclosure. The drivers 804a through 804d are configured for operation with one or more types of data sources, such as a data file produced using a particular database program. Depending on the characteristics of the database program, it may be necessary to have, e.g., two separate drivers for data files produced by two separate versions of a database software package. In some circumstances it may be possible to provide a single driver which can be used in connection with data files produced using two (or more) different types or brands of software (or different versions of a given brand of database or other software).

The source data 806a through 806d depicted in Fig. 8 may, in general, be any computer readable information source. Examples include flat file source data, hierarchical databases, relational databases, spreadsheets, and the like. Although Fig. 8 depicts an embodiment in which four data sources are shown, the present invention could be used in context in which there was only a single data source, or in which there were five or more data sources. Although the present invention may be used in situations in which each data source 806a through 806d is produced using a different type or brand of software, it is also possible to use the present invention in situations in which two or more of the data sources are produced by the same brand or type of database or other software. As one example, the first driver 804a might be configured for use in retrieving information from data files that were produced using dBase II®, the second driver might be configured for use in retrieving information from data files that were produced using dBaseIII®, the third driver might be configured for use in retrieving information from data files that were produced using a flat-file system such as Simply Accounting™ and the fourth driver might be configured for use in retrieving information from data files that were produced using Microsoft Access®.

Once the data source or sources have been analyzed (as described more thoroughly below), the results of such analysis can be used in a number of ways, including providing for user-access to the information in the data sources for viewing or editing, copying some or all of the data and, preferably enhancements thereof (as described below) to create a new database, creating data reports (for viewing, printing, storage, transmission, etc.), queries and the like.

In the embodiment of Fig. 8, after the main process 802, using the drivers 804, performs an analysis of the source data 806a through 806d, it may create one or more new databases 808, containing data from one or more of the various data sources 806a through 806d. In one embodiment, one new database is created for each data source. It may be desired to join two or more such databases, e.g. using standard database techniques, such as when such databases have similar structure. In another embodiment, one database 806 may contain information from two or more data sources (e.g. if a company uses one database or other data source for storing sales information and another database or other data source for storing employee information). The new database or databases 808 may, if desired, be used for generating reports, e.g., using a report writer 810, and may, if desired, be used for entering, viewing or analyzing data, e.g., using a database management system 812 or other software 814. In one embodiment, the database 808 is a Microsoft® Access database which includes base code having one or more

wizards, templates, filters and or tool kit software (as those terms are understood by those skilled in Microsoft® Access programming) e.g. for providing database reporting and analysis such as outputting standard financial reports. In one embodiment, financial and management reporting software is provided as an extension or modification of that available under the tradename MV™ available from Timeline, Inc. of Bellevue, WA.

Preferably, the analysis system includes a module configured for generating or providing reports or screens for specific purposes or groups. For example, in the embodiment of Fig. 8, an executive information system (EIS) 815 is provided preferably with an easy-to-use user interface and configured to output analysis of the data in various spreadsheet, presentation graphics, or printed forms, preferably using navigation tools or graphic selectors (e.g., to select between bar charts and pie charts, to select reporting periods, to select quarterly v. weekly reports and the like). In one embodiment, the executive is presented with a menu of various views of the data, including views which may have been automatically generated or detected using the automatic roll-up generation procedure as described above.

In one embodiment the information in the new database or databases 808 is enhanced, e.g. by main process 802 in the sense that it is configured to generate, display or output analysis or relationships of data which were not displayed or output by or using the source data 806a through 806d

Fig. 9 is a schematic depiction of various drivers 804a through 804d. Each driver includes a plurality of defined processes or functions 901 through 909. Each function may include computer program instructions 912, e.g., to implement and carry out one or more of the steps described below and depicted in Fig. 10. In one embodiment, each function 901 - 909 is a callable subroutine or procedure. The functions 901 through 909 defined in a given driver 804b include functions which must be performed or carried out differently depending on the type of source data 806a, 806b. Thus, for example, with regard to function one 901, which is a function designed to select certain directories on a hard disk or other information storage device where desired information is stored, the procedure for selecting directories will differ depending on the type of source data 806, as can be seen, e.g., from a comparison of the example of Fig. 2 with the example of Fig. 5. Accordingly, the programming 912 which implements function one in the first driver 804a may be different from the programming code which implements a corresponding function in the second driver 804b. In this way, each driver defines one or more processes for performing a function with such procedure being configured to accommodate the different characteristics of two or more different types of source data. For example, Fig. 11A depicts a portion of a procedure, expressed in pseudo-code, of a type which might be used in connection with selecting and/or searching directories in connection with the directory structure depicted in Fig. 2, while Fig. 11B shows corresponding portions of pseudo-code for a procedure which might be used in connection with selecting or searching directories for the directory structure depicted in Fig. 5. Those of skill in the art will understand, from the examples of Figs. 11A and 11B, how to configure drivers to perform the same function on two different types of source data. Although Fig. 9 depicts a function module having nine functions, the present invention could be used in connection with a function module having more or fewer functions. It is possible to configure a system in which different function modules define different numbers of functions and/or in which one or more functions are configured to provide or return a null value or a constant value or information.

A number of procedures can be used to start 1002 the method which is depicted in Fig. 10. In one embodiment, the method of Fig. 10 is implemented using a computer program stored on a medium such as a hard disk, CD-ROM or other non-volatile medium, and the method is started by issuing instructions to the computer (e.g. via a keyboard, mouse, etc.) to launch the program, i.e., to load the program into memory and execute the program. Alternatively, the program may be launched by another program. For example, in one embodiment, the new database 808 is a Microsoft® Access Database which can include a routine, such as a so-called "wizard," for launching the program, with the program (Fig. 10), in turn, accessing data in the information sources 806 to populate or update the database 808. In this embodiment, it may be useful to employ the wizard to display prompts or "dialog boxes" for the purpose of soliciting user input as needed (e.g. for step 1020), so that the user interface will have an appearance which is consistent with the user interface for the database 808.

In the procedure depicted in Fig. 10, the first step after the procedure starts 1002 is to identify and initialize the dynamic drivers 1004. In this context, the drivers 804 are considered to be dynamic in the sense that drivers can be modularly added or deleted, e.g., to accommodate a new or different type of data source. For example, a user may initially be provided with a system such as that depicted in Fig. 8 having four drivers, but may at some future time add additional drivers through purchase from a software vendor, downloading from an information service, network, Internet connection and the like, or by writing a custom driver. Because of the modular and dynamic nature of the drivers, it will not be known, in advance, which drivers are available, and thus, when the program is started 1002, the program identifies the drivers that are available to it. In one embodiment, this is performed by searching a disk or a directory thereof for files having a predetermined (partial) file name or file extension. In one embodiment, the program may further analyze selected portions of each file, e.g., header information, to verify that files identified by such file name and/or extension are desired drivers. Initializing the drivers generally includes identifying and linking the driver functions and initializing data within each driver.

It is then determined whether the procedure will perform an import or an update 1006. In an import, a procedure is first done in which all or a majority of the data and structure in the data source is accessed and saved to the new databases. In an update, a procedure is done in which only a selected portion of the data and/or structure are accessed, e.g., to assure that the information in new databases 808 reflects recent changes or additions that may have been made to the data source 806. In a typical situation, an import will be performed the first time the system of Fig. 8 accesses or uses information from a given data source, or if relatively major changes or additions have been made to the data source. Typically, an update will be performed on a regular basis (e.g., daily, weekly, etc.) in order to synchronize the data in the source data 806 with the data in the new databases 808. In one embodiment, the selection to import or update 1006 is performed automatically, i.e., by performing an update unless this is the first time the procedure has accessed the particular data source. In another embodiment, a user is permitted to choose between import and update by providing input, e.g., via a keyboard selection, using a pointer device, and the like.

If an import has been selected 1008, the main procedure 802 will initiate the execution of a function of one or more of the drivers to select directories to search 1010. The driver 804 which is loaded or called by the main process 802 will depend on which type of source data is being accessed. In particular, for a given data source 806a, the main process 802 will employ the driver 804b, which is configured to accommodate that type of source data 806a. If more than one data source 806 is to be accessed, the main process 802 will use whichever driver 804 is configured for each type of source data 806. Preferably, the type of the data source is determined automatically

based on characteristics such as the names (or "extensions") of the files and/or directories, the number, size and structure of the files, header or other information in the files. In another embodiment, a user is permitted or requested to indicate the type of data source (e.g. by identifying the brand name and version number of the software that was used to create the data source files or by indicating whether the user wishes the procedure to search only local disk files or to perform a search that includes network files).

At the end of step 1010, the main process 802 will have access to a stored list of the directories which are to be searched for all data sources 806 as identified by drivers 804 which were identified in step 1004. After step 1010, the main process 802 loads or activates or calls another function 902 of the function modules 804a through 804d, in order to search the directories which were selected in step 1010 for the data which is to be imported 1012. The search of directories 1012 is performed in a manner which will depend on how the information is stored in different data sources 806. For example, for some types of source data it may be sufficient to identify files having a certain file name and/or certain file extension. For other types of source data 806 it may be necessary to scan the data in various files to identify files which have certain structures or contents, e.g., in a header portion of a file, or elsewhere. Thus, different modules 804 will be configured to provide the "search directories" function 902 in different manners, to accommodate different data sources 806.

If an update 1014 rather than an import 1008 is to be performed, it is not necessary to select and search directories, since, preferably, when an import is initially performed, the results of the functions 901, 902 for selecting and searching directories are stored in a manner which can be accessed by the main process 802 at a later time. Thus, using such stored information the main process 802 is able to identify data which was previously imported or updated. In one embodiment, this is useful to prevent loading of redundant data, i.e., data which is already present in the new databases 808. In general, for a full update it is desired at step 1016 to identify data which is new or changed since the last import or update such that at least some data already in the databases 808 will not be re-loaded. In one embodiment, in order to prevent redundant data loads, the system will attempt to identify data which has not changed since the last import or update. In general, if this procedure is followed, at the end of the procedure the data in the new databases 808 will be synchronized with the information in the source data 806, i.e., it will contain information which accurately indicates the structure and data of the data sources 806 in their current state.

In the embodiment of Fig. 10, an identification of the data which is to be imported or updated is displayed 1018 in order to give the user an opportunity to select the data to be imported or updated or to choose to refrain from importing or updating certain data. The manner in which the indication of data to be updated or imported is organized for display will depend on what type of data source is being accessed, and thus will be provided in response to a call or activation of a function in one of the drivers 804 (902a). For example, a function 902a of a driver which is configured for use in connection with the data source depicted in Figs. 4A through 4F might display a list of the subsidiary companies drawn from table 430 (Fig. 4F) in order to give the user an opportunity to import or update data for some companies but not for others. Preferably, the user may select one or more companies from a displayed list. The display in selection steps 1018, 1020 can be repeated for other types of organizations or data in the data source, e.g., to display and permit selection of data specific to certain employee 416, to certain accounts 412, depending on how the display function 902a of the driver is written or configured.

As will be described more thoroughly below, in some cases, after the new database 808 is created, the user will continue to use the original information sources 806a - 806d for creating, storing, editing and, in some cases, viewing, data. Thus, the present invention is compatible with situations in which the user wishes to continue to use the databases 806, user interfaces and the like, with which the user is already familiar. In this case, however, it is desirable to update the new database 808, from time to time, to reflect new or edited data in the information sources 806a - 806d which have occurred since the new database 808 was first created (or last updated). One fashion of performing such updates would be to repeat the entire process of creating the database 808 (described below). In many situations, however, this would be inefficient since a large portion of the originally-created new database 808 is unchanged and still valid. Thus, according to one embodiment of the present invention, procedures are provided which permit the new database 808 to be updated with only that data from the information sources 806a - 806d which have changed since the last update of the database 808.

The frequency with which the new database 808 must be updated depends upon the frequency with which various uses are made of the database 808. Thus, it may make little sense to update the database 808 on a daily basis if it is accessed only once per week. Moreover, the particular information included in the update may vary according to the use made of the new database 808. Thus, if factory production data is reviewed by management only on a quarterly basis, there is no need to include such data in the weekly updates which may be used, e.g., for reviewing regional sales. Preferably, one or more profiles defining various types of updates are created and stored and, preferably, a scheduler is also created and stored. The scheduler is used to define the times at which to automatically execute such updates (i.e., without requiring a person to "calendar" or remember to initiate such updates). Preferably, different update profiles are scheduled to be executed at different times or intervals, such as by providing a scheduler which automatically executes, e.g., regional sales update procedures weekly before a weekly sales meeting and which schedules quarterly updates of production data prior to quarterly production reviews.

Because different persons utilizing the system of Fig. 8 may use different components of the system (e.g., information sources 806a - 806d or new database 808) for day-to-day operations, the profiles may be stored along with the new database 822, the old database 824a - 824d (if there is appropriate "write permission" for storing information in the information sources 806a - 806d) or both.

Various schemes can be used to create profiles. It is possible to include commands (such as menu selections or buttons) to allow a user to request creation of an update profile (e.g., via a predetermined series of screen forms or dialog boxes, such as provided in Microsoft® Access Wizards). It is also possible to automatically prompt a user for a decision regarding creation or editing of a profile, e.g., whenever a user requests a screen or printed report which uses data not currently in a defined profile (or which uses data more frequently than the current profile update frequency for that data). In one embodiment, the profile 822 is created by creating a database table which lists each field of the database 808 and, for each such field, indicates whether there is an existing profile for updating that database, the identity or name of such profile and the current update frequency or interval.

In addition to automatically-scheduled updates, the user of the system of Fig. 8 is preferably also provided with an option to request an update at any time, preferably with the option to select among any of the predefined update profiles or request a full update. If desired, the system may prompt the user for a decision regarding whether this set of data to be updated should be stored as a profile (as described above). The system may also provide a list

of already-existing or defined profiles so that the user may select among predefined profiles rather than selecting among displayed data (such as a list of subsidiary companies).

In some cases, it may be determined beforehand that it is desired to always import or update all available information from the data source ("full update"), and thus the function 902a for the applied driver can merely return program flow to the main process 802. For example, with respect to the data source depicted in Figs. 7A through 7D, it may be determined that all available surface temperature data from all sites are to be always included in every import or update. In one embodiment, the user may be provided with a display indicating the location of the data which corresponds to the user's selection, such as a display of the directories, sub-directories and files containing the information, and may optionally be given an opportunity to select which directories, files, etc. are to be accessed.

Once it has been determined, e.g., by steps 1018 and 1020, the data which is to be imported or updated, general information is loaded 1022. If it is desired to access information from two or more data sources, this can be done either serially (i.e., performing steps 1022 through 1046 on a first data source using a first module, followed by performing steps 1022 through 1046 on a second data source using the appropriate driver, and so on) or in parallel (i.e., performing steps on each desired data source using the appropriate drivers before performing later steps on each information module).

General information includes information about the structure of the data in the data source. The type of general information which is loaded in this step 1022 will vary for various types of source data. For example, for a function 903 which is written or configured for use in connection with a database such as that depicted in Figs. 4A through 4F, the general information may include, e.g. an identification of accounts or other categories used in the data source 806. On the other hand, if the function 903 of a driver 804 is configured or written for use in connection with the data source depicted in Fig. 1, it may still be desired to determine how many components are used in the data source but, in this case, this information will be determined by the number of flat files 101a through 101f found in the data source 806. General information may also include information such as how many projects 112, how many products 114 and/or product lines 116, or how many subsidiaries 118, are defined in the data source 806. General information can also include company name, first month of the fiscal year and, in general, any other information that can be loaded once (as opposed to, e.g. information loaded in steps 1024, 1033 and 1036, typically loaded in a loop). If a "load general information" function 903 is provided in a driver configured for use in connection with the data source depicted in Fig. 6, general information such as the number of locations 612 in the database might be loaded in step 1022.

The main procedure 802 also calls or activates a function 904 of the appropriate driver or modules 804 to load data definitions 1024. Data definition can include information such as the text name stored as an identifier for a particular class or category of data in the data source 806, the field size, type of data (string, integer or decimal; number of decimal places) and similar characteristics for various categories of data. Preferably, loading of the data definition includes interrogation of the data to obtain information necessary to store an indication of the architecture or structure of the information in the data source and the data elements in the data source as required to generate one or more new databases 808 which will contain all of the structure and data needed for the type of reporting or analysis to be performed on the new database. The interrogation of the data in the "load data definition" step is intelligent interrogation in the sense that it can conform to virtually any data source and identify what is required to

store a standard form of the data source, e.g. for reporting and analysis. In the example of Figs. 4A through 4F, information needed to indicate the architecture of the source data would include, e.g., the names of the four account parts (Account, Company, Employee and Location) as well as the data type (e.g. numeric or string) and length required to store any string account parts. In the example of Fig. 6, information needed to indicate the architecture of that data source would include storing the names of the account parts (Location and Date) as well as the names of the references used for this data (Unit). Interrogation may include identifying other optional data that can be loaded, e.g. invoice numbers. The particular type of interrogation performed will depend on the characteristics of the particular data source being analyzed and thus is different for each driver 804. In general, steps of loading data or information 1022, 1024, 1033, 1036 are performed by functions in drivers 804, while steps of saving information 1026, 1028, 1030, 1032, 1034, 1038 are performed by the main process 802.

The main process 802 then identifies or creates a database repository 1026, i.e., identifies or creates a file or other data storage structure in the new databases 808 which will serve as the location where information loaded from the data source or sources 806 will be saved. An update typically does not need to create a new database or database tables, since updates usually simply add to already-existing tables within an already-existing database.

The main procedure 802 then calls the appropriate function 905 of one or more drivers to create database tables 1028 which will be used for storing data saved from the source data into the new databases 808. The manner in which the database tables are created preferably takes into account both the structure of the data and the data source or sources 806 and the manner in which the new databases 808 will be used, e.g., for analysis, generating reports, etc. Because the particular tables which are created will vary depending on the characteristics of the information in the data source 806 (as determined, e.g., by steps 1022 and 1024), the create database tables 1028 are functions provided by drivers 804 configured for the particular database being used for access. For example, when the "create database table" function 905 is written or configured for use in connection with an data source such as that depicted in Figs. 4A through 4F, the database tables which are created may include, e.g., an Account table, a employee table, a subsidiary table (which may be a roll up of Account), and a details table (as described more thoroughly below), whereas a "create database table" function written or provided in a driver configured for use in connection with the database depicted in Figs. 7A through 7D may include a location table, a date table, a time table, a unit table and a detail table. Preferably, the tables created in the new databases 808 have a structure or architecture which is dynamic in the sense that it can accommodate virtually any type of data definition or structure which might be found in various data sources 806. In one embodiment, the new databases 808 are intended to be used primarily for outputting information such as generating reports and analysis and thus are preferably configured, as described below, to provide superior output performance such as high flexibility in the types of outputs and data analysis available and relatively rapid execution of such analysis and/or output. In this context a database is optimized for speed and/or flexibility of output if it provides speed or flexibility of output which is superior to the speed or flexibility of some other possible configuration. Thus, in this context, "optimized" does not necessarily require a mathematically precise optimization. In one embodiment, three general types of tables are provided in step 1028: a plurality of category tables (including rollup tables where appropriate), at least one detail table and at least one entry table. Preferably, a category table is provided for each manner in which a particular data point or record can be categorized. For example, if the "create database table" function 905 is provided in a driver 804 which is configured for use with the database as depicted in Figs. 4A through 4F, the new database, as depicted

in Fig. 12, will contain a plurality of category tables 1202 including, e.g., an Account table which lists all the possible account categories of account 1203, a subsidiary table 1230 which lists all the possible subsidiaries found in the data source 806, a product table 1232 which lists all of the products noted in the data source 806, a employee table 1216 which lists all the employee noted in the data source 806, and a location table 1234 listing various locations, sales regions, etc., noted in the data source 806. In the depicted embodiment, each record or item in each of the category tables 1202 is associated with an index for use in a detail table as described below.

In the embodiment depicted in Fig. 12, the detail table 1240 will be provided that will, once populated, have a record for each accounting entry or transaction in the data source or sources 806. In the embodiment of Fig. 12 for fields 1244, namely a date field 1242b, description field 1242c, quantity field 1242d, and amount field 1242e are provided corresponding to a transaction in the data source (Fig. 4A). An index field 1242a is provided for storing an identifier number or index number for each record. Additionally, for each record a separate field is provided for storing an indication of any appropriate information for each of the categories defined in the load general information step 1022, in the example of Fig. 12, including the Account category 1242f, the subsidiary category 1242g, the product category 1242h, and the employee category 1242i. In general, it is desired to provide as many different fields, i.e., categories, which are present in the data source 806 as are needed for analyzing or outputting data. Thus, since it might be desired to output a report which groups the transactions according to which Account the transaction relates to, it is useful to have an Account category. Further, it might be that in the desired accounting purpose, it is desired to print out a separate report for each subsidiary or to print out a report in which transactions are grouped by subsidiary, thus it is useful to have a subsidiary category 1242g. In general, for each desired way of selecting, grouping, reporting, printing or analyzing the data, a separate field may be provided in the detail table 1240.

The structure of the database depicted in Fig. 12 may be contrasted with the structure of the data source depicted in Figs. 4A through 4F and the structure of the data source depicted in Fig. 1. For example, in the structure depicted in Figs. 4A through 4F, the manner in which a particular transaction (Fig. 4A) was associated with a particular account (Fig. 4B) was indicated in a separate link table (Fig. 4D), whereas in the embodiment of Fig. 12 the index for the appropriate account 1242f will be stored in its own field of the same record which contains the transaction information 1244. Thus, although the database 808 having tables as depicted in Fig. 12 can store the information found in either an data source as depicted in Fig. 1 or an data source as depicted in Figs. 4A through 4F (or source data structured in other ways), the structure or architecture of the database in the example of Fig. 12 is different from that of either the data source depicted in Fig. 1 or the data source depicted in Figs. 4A through 4F. Similarly, the relational database structure of Fig. 12 is different from the flat file structure depicted in Fig. 1 even though the type of information stored in the two organizations is similar.

The main procedure 802 can be configured to save the general information 1030 (loaded in step 1022) and save the data definition 1032 (loaded in step 1024), e.g., in additional tables provided in the databases 808, e.g., for use in later steps of Fig. 10 and/or in updates performed at a later time.

The main procedure 802 uses appropriate functions 908 of one or more drivers 804, to load into the new databases 808, data definition codes (e.g., field width, data type, etc.) for the various tables created in step 1028 (1033). In one embodiment, rollup information is also loaded at this time. In general, rollup information refers to information used for defining sub-categories of data, i.e., groups of items within one of the category tables. As an

example, as depicted in Fig. 12, the employee category table may be associated with a location or region code, e.g., to identify the location or site or region where each company employee is located or has responsibility. For example, there may be numerous sales employee for a company, each of which is associated with a sales region. As another example, the various products of a company may be rolled up into or grouped into product lines. By defining a field 1238 for a location rollup code, the structure of Fig. 12 makes it possible to output a report which is grouped by sales region. Rollups may also be used for providing statistical analysis of group data such as averages, means, standard deviations, etc. Although in the embodiment of Fig. 12, the field for the location rollup code 1238 is shown as being a field of the employee category table 1216, a location field 1238 could also be provided in the detail table if desired.

The manner in which a function for loading a rollup code 908 operates will vary depending on the type of source data 806 with which it is configured to act, and thus it is provided as a function of various drivers 804 so that different programming instructions can be provided for use with different types of source data. As one example, a "define rollup code" function 908 may be provided in a driver 804 configured for use in connection with the database depicted in Figs. 4A through 4F. In this example, a location field is already defined in the employee table 416 which can be directly used for location code purposes. In contrast, in the embodiment depicted in Fig. 1, there is no indication of location for an employee 120 associated with a particular flat file 101a. Thus, in one embodiment, a location rollup may not be possible for data obtained from the data source in Fig. 1. If, however, there is, e.g., another file which provides the home address for each employee in the company, it may be possible to use, e.g., the state of residence of each salesperson to infer the sales region that person is responsible for and thus define a location rollup code inferentially. Additionally, a "define rollup code" function may include access to information which can be used for defining a rollup code. For example, in connection with the data source depicted in Fig. 6, the "define data rollup" function 908 may contain, within that function, a table indicating, for each potential temperature station 612, whether that station is a northern hemisphere or a southern hemisphere station and thus could create a hemisphere rollup code on such basis. In some cases it may be desirable to provide for word recognition and/or searching for key words in the database in order to define additional rollups and/or structures.

In some situations, the rollup code will be relative to information which was not used in the data source as a basis for analyzing or grouping data (e.g., the hemisphere rollup code for the data source of Fig. 6). Thus, in these situations providing a rollup code involves providing an enhancement of the data by automatically providing additional elements which were not available (or at least not used) for outputting or analyzing information in the data source 806. Preferably, data is classified into categories and then grouped by an in-depth analysis of the data source's data.

The system may be configured to recognize and add certain reporting relationships which can be determined with a degree of definiteness depending on the data in the information sources 806a - 806d or the structure of that data. For example, sales information may have sales which are organized by sales districts (either expressly in the information sources 806a - 806d or inferred, e.g., using salesman addresses as described above). These districts may be further viewed or analyzed, according to an embodiment of the present invention, by other groupings such as by two or more vice presidents, each of which may be responsible for two or more sales districts, provided the relationship of the vice president to the sales districts can be found expressly in or inferred from the information sources 806a - 806d.

Preferably, such additional roll-ups or enhancements are defined only if the data indicates it may be meaningful to view the data according to the roll-up criteria. For example, in one embodiment, an enhancement or additional relationship is not automatically added unless there exists more than one value for a particular field or parameter but, preferably, fewer than the number of records containing that field or parameter. For example, if there are at least two different vice presidents responsible for different districts, then it may be meaningful to look at the sales according to the amount of sales corresponding to each vice president. However, in the example of Figs. 6 -7, if all reporting stations were located in tropical areas, the system would not automatically create a roll-up or enhancement to view weather data according to categories of snowfall amounts, since all stations will report zero for snowfall amounts in this example. Thus, in a business database, the system will preferably examine the data in the source databases 806a - 806d to determine whether, for example, there is more than one salesperson, more than one region and/or more than one product. If there is more than one value in a field for the various records of a database, the system can be configured to automatically provide an option to view the data according to that field, e.g., grouped by the average or sum of values (such as average or sum of sales, expenses, etc.). In one embodiment, this is achieved by creating a Microsoft Excel® "pivot" table.

As one example, referring to Figs. 4A - 4E, the table of Fig. 4D can be used to determine, for any transaction, which employee is associated with that transaction. Table 4C can be used to determine whether, for a given employee, that employee is associated with the Northwest region, Southeast region or the Central region. If all transactions are associated with employees in the Central region, there is no point in displaying transactions by region. However, if the transactions in table 4D are associated with employees in at least two different regions, the system can be configured to automatically generate a roll-up which displays the average or the sum of transactions (and/or transactions of a particular account type as determined from Fig. 4B) broken-down by the location or region associated with the employee who is associated with that transaction.

In addition to defining rollups, the process may also store optional reference fields. In general, optional reference fields refer to fields which will typically not be used for grouping data such as free-text fields (comments, memo fields, invoice numbers, etc.) but which may be desired for inclusion in reports, etc.

Following loading of data definition codes and rollups, these data definition codes and rollups are saved 1034 in the new databases 808, e.g., by listing the categories into the various category tables 1202. Loading and saving data definition and rollup code 1033, 1034 is performed in a loop fashion 1035 to load and save the specific categories (specific Accounts, Companies, Employees and Locations, in the example of Figs 4A-4F).

In review, according to the depicted embodiment, step 1024 defines the categories (e.g. Account, Company, Employee and Location), as well as their data types (e.g. string or numeric). Step 1028 creates the category tables defined in step 1024 (and any other tables defined in step 1022). Step 1032 saves the data definition, typically in one standard table. Steps 1033 and 1034 load and save data definitions and rollup codes.

At this point in the process, although information relating to the structure of the data has been placed in the new databases 808, data which is the main subject of the source data 806 (e.g., accounting entries or transaction in the case of accounting source data, temperature data in the case of meteorological temperature source data) has not been loaded into the new databases 808. Accordingly, the main process 802 calls or activates a function 909 in the appropriate drivers 804 to load data 1036, save the data 1038, and repeat the process 1039 until all the desired data

has been loaded and saved 1040. Thus at the end of this procedure 1036, 1038, 1040, the new databases 808 will be populated with data from one or more data sources 806.

When two or more data sources having different structures and/or produced using different brands or types of software are combined, by using the procedure of Fig. 10, these data from different types of sources can be populated into a common database structure, e.g., as depicted in Fig. 12. This facilitates common or standardized analysis and reporting for the data, preferably optimized to provide flexibility and speed of output.

In the embodiment of Fig. 10, the main process 802 can now build and, if desired, execute, data queries such as summarization queries 1042. In general, there are at least three types of queries which may be built. The first type of query may be provided which will be common to all new databases 808 which are created using the procedure of Fig. 10, such as a query which provides the number of entries in the detail table or the number of entries for a given date range (e.g., per quarter). Other queries may be built at least partially depending on the general information and data definition obtained with respect to one or more of the data sources 806, including rollups which may have been provided and accordingly could, if desired, be provided as part of a driver 804, specific to a particular data source. A third type of query may be provided to replicate or include queries or reports which were used in the original data source (e.g., as depicted in Fig. 4E).

After the new database 808 is populated, the system, according to one embodiment, will perform certain audits or checks of the data, automatically, to verify that the system operated properly. Various types of checks can be included. Samples of the data in the new database 808 (or, if desired, all of the data) can be compared with corresponding data in the source 806a - 806d to verify that data was not corrupted. The new database 808 can be checked to verify that necessary structures are present. For example, in the case of an accounting database, the new database 808 can be automatically checked to determine that the required account parts are present. The new database 808 can be checked for empty sets, such as by making sure that rollups, or other enhancements that have been defined, are non-empty.

Once a new database 808 is populated and verified and appropriate queries are built, the main procedure 802 may close the tables and databases 1044 and the dynamic drivers 1046, e.g., to free up memory.

Although a major contemplated use of the present invention is in connection with providing standardized and/or enhanced reporting and analysis of data in one or more data sources, it is also possible to use the present invention in connection with data entry and data storage by using a database management system (e.g., Microsoft Access®, Excel®, FoxPro®, Btrieve®, etc.) in connection with the new databases 808. Although it is contemplated that a major use of the present invention will involve continuing to employ the original source data 806 for inputting and storage while maintaining a copy of the same information in the new databases 808 for reporting and analysis purposes, it is also possible to use the present invention for transferring data from one type of source data 806a, b, to another type, e.g., 806c, by first storing it in the new databases 808, as described above and, thereafter, downloading or exporting the information from the new databases 808 to a different type of source data 806c.

In light of the above description, a number of advantages of the present invention can be seen. The present invention facilitates standardization of reporting and analysis despite a variety in the brands or types of data sources used. The present invention provides for a system which can be optimized or otherwise provide improved performance in outputting or reporting data. The present invention provides for data reporting and analysis capability which is enhanced compared to data reporting and analysis of the data sources. Via sophisticated

interrogation of the source data, in the context of an accounting system the present invention is able to mirror the chart of accounts setup in the data source. In one embodiment, the process extracts some or all defined rollups, optional reference fields, accounting period information. By performing the tasks automatically and eliminating the need for human analysis (or reducing), the present invention is less labor-intensive and less time-intensive than previous methods, in some cases making it possible to populate new databases 808 in a matter of a few minutes or hours in situations that would require several days or weeks under previous methods. In one embodiment, the driver 804 can be configured to detect, analyze and maintain, in the new databases 808, any security, passwords, permissions, etc., that are used in the data source 806. In this way it is not necessary for a systems administrator to need to maintain a new and separate set of accounts, passwords, permissions, etc., for the new databases 808 in addition to that maintained with the original data source 806. Preferably, the system can be configured to perform updates substantially at predetermined intervals such as daily, weekly, etc.

A number of variations and modifications of the present invention can also be used. It is possible to use some aspects of the invention without using others. For example, it is possible to provide for populating a new database 808 without defining new or additional rollups. Although in the above description, the various drivers 804 can be provided as separate DLL files and are dynamic in the sense that as many as desired can be added simply by storing additional DLL files in the proper directory, it would also be possible to make an operable version of the invention in which the function performed by the function modules are provided as portions of or subroutines called by the main procedure 802 rather than being separately stored modules.

While the invention has been described by way of a preferred embodiment in certain variations and modifications, other variations and modifications can also be used, the invention being defined by the following claims:

WHAT IS CLAIMED IS:

1. A computer-implemented method comprising;

providing a first driver which issues instructions for accessing data which may be stored in either of first and second different data sources, said first driver containing program instructions configured for use in connection with said first data source; and

using said first driver to automatically obtain information about the data structure of said first data source without the need for human analysis of the first data source.

2. A computer-implemented method usable in connection with accessing data which may be stored in either of first and second different data sources, the method comprising:

a first step for providing a first driver containing program instructions configured for use in connection with said first data source;

a second step for using said first driver to automatically obtain information about the data structure of said first data source without the need for human analysis of the first data source.

3. A method, as claimed in claim 2, further comprising:

using said first information to define a structure for a first database.

4. A method, as claimed in claim 2, further comprising:

using said first driver to store at least some information from said first data source into a first database.

5. A method, as claimed in claim 4, wherein said first database is enhanced with respect to said first data source.

6. A method, as claimed in claim 2, wherein said first driver includes programming code which is callable by a main procedure and wherein said method is extensible for accommodation of said second data source by providing and calling a second driver containing program instructions configured for use in connection with said second data source, in the absence of substantial modification of said main procedure.

7. A method, as claimed in claim 2, wherein said method is flexible to permit accessing any readable data source by providing a driver configured for use with such readable data source.

8. A computer-implemented method usable in connection with accessing data which may be stored in either of first and second different data sources, at least one of said first and second data sources configured to generate at least a first output, the method comprising:

providing a first driver containing program instructions configured for use in connection with said first data source;

providing a second driver, different from said first driver, containing programming code configured for use in connection with said second data source;

using said first and second drivers to obtain first and second information about the data structure of said first and second data sources, respectively;

using said first and second information to define a structure for a first database;

using said first and second driver to store at least some information from said first and second data sources into said first database.

9. A method as claimed in claim 8, further comprising:

generating at least a first report based on the information in said first database

10. A method, as claimed in claim 9, wherein said first report is enhanced with respect to said first output.

11. A method, as claimed in claim 8, wherein each driver includes program instructions for performing a plurality of functions.

12. A method, as claimed in claim 11, wherein said plurality of functions includes at least one function selected from among:

selecting directories where said data sources are located;

searching directories for data files;

displaying data to be input or updated;

loading general information from said data sources;

loading data definitions from said data sources;

creating database tables configured for storing at least some information from said data sources;

saving general information obtained from said data sources;

saving data definitions from said data sources;

loading data definitions codes into said first database;

loading rollup information into said first database; and

loading data from said data sources into said first database.

13. A method, as claimed in claim 8, wherein said first database is optimized for speed of data output.

14. A method, as claimed in claim 8, wherein said first database is optimized for flexibility of data output.

15. A method, as claimed in claim 8, wherein said first database includes a plurality of category tables, at least one detail table and at least one entry table.

16. Apparatus usable in connection with accessing data which may be stored in either of first and second different data sources, at least one of said first and second data sources configured to generate at least a first output, the apparatus comprising a computer coupled to said first and second data sources and programmed for:

providing a first driver containing program instructions configured for use in connection with said first data source;

providing a second driver, different from said first driver, containing programming code configured for use in connection with said second data source;

using said first and second drivers to obtain first and second information about the data structure of said first and second data sources, respectively;

using said first and second information to define a structure for a first database;

using said first and second driver to store at least some information from said first and second data sources into said first database.

17. Apparatus usable in connection with accessing data which may be stored in either of first and second different data sources, the apparatus comprising:

a first driver means containing program instructions configured for use in connection with said first data source;

a second driver means, different from said first driver means , containing programming code configured for use in connection with said second data source;

means for using said first and second driver means to obtain first and second information about the data structure of said first and second data sources, respectively;

means for using said first and second information to define a structure for a first database;

means for using said first and second driver means to store at least some information from said first and second data sources into said first database.

18. A computer-readable medium, having stored therein a computer program usable in connection with accessing data which may be stored in either of first and second different data sources, at least one of said first and second data sources configured to generate at least a first output, the computer program comprising instructions for:

providing a first driver containing program instructions configured for use in connection with said first data source;

providing a second driver, different from said first driver, containing programming code configured for use in connection with said second data source;

using said first and second drivers to obtain first and second information about the data structure of said first and second data sources, respectively;

using said first and second information to define a structure for a first database;

using said first and second driver to store at least some information from said first and second data sources into said first database.

19. A computer-implemented method comprising the steps of:

providing a first driver which issues instructions for accessing data stored in a first data source;

using said first driver to obtain first information about the data structure of said first data source;

using said first information to define a structure for a first database;

using said first driver to store at least some information from said first data source into said first database;

storing second information defining at least portions of said data stored in said first data source to be used in updating said first database; and

updating said first database, using said second information.

20. A method, as claimed in claim 8, wherein at least one of said first and second data sources includes a plurality of records, each having a plurality of fields for storing data values, and further comprising:

identifying at least one field wherein at least first and second different data values are stored in said one field of at least two records; and

displaying first and second data groupings corresponding to said first and second different data values.

21. A method, as claimed in claim 8, further comprising analyzing information stored in said first database to identify errors.